UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/537,571	06/03/2005	Hoi Yeung Chan	YOR920020206US1 (8728-587	4559

46069 7590 09/29/2006

F. CHAU & ASSOCIATES, LLC
130 WOODBURY ROAD
WOODBURY, NY 11797

EXAMINER	
BROWN JR, NATHAN H	

ART UNIT	PAPER NUMBER
2121	

DATE MAILED: 09/29/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/537,571	CHAN ET AL.	
	Examiner	Art Unit	
	Nathan H. Brown, Jr.	2121	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *(3)* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 June 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 June 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Examiner's Detailed Office Action

1. This Office is responsive to application 10/537,571, filed June 3, 2005.

2. Claims 1-33 have been examined.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

4. Claims 1-33 are rejected under 35 U.S.C. 102(a) as being anticipated by *IBM*, "WebSphere Application Server Enterprise Services Business Rule Beans (BRBeans)", 2001.

Regarding claim 1. *IBM* teaches a method for deploying computer infrastructure, comprising integrating computer-readable code into a computing system (*see* p. 2, §Business Rule Beans (BRBeans), "The Business Rule Beans (BRBeans) framework extends the scope of Websphere Application Server Enterprise Edition", *Examiner interprets extending the "Websphere Application Server" to be a method for deploying computer infrastructure, comprising integrating computer-readable code into a computing system.*), wherein the code in combination

with the computing system is capable of performing: identifying inferencing aspects for a program (*see* p. 2, §What is a business rule?, "A business rule is a statement that defines or constrains some aspect of a business ... At its simplest level, a business rule is little more than a well placed IF/THEN statement that compares a variable against a determined value, and then issues a command when they match.", *Examiner interprets the operation of an IF/THEN statement to be a modus ponens inference.*); and providing the identified inferencing aspects as inferencing components (*see* pp. 17-18, §The BRBeans framework – overview), wherein the inferencing components are externalizable (see p. 32, §Externalized business rules, *Examiner interprets the objects Rule and RuleImplementor to be externalized components.*).

Regarding claim 24. *IBM* teaches the system (see p. 2, §Business Rule Beans (BRBeans), "The Business Rule Beans (BRBeans) framework", *Examiner interprets "BRBeans to be a software system.*), comprising: an identification component configured to identify at least one point of variability within an application program (*see* p. 2, "Rule externalization is accomplished by extending the application analysis and design processes to identify the points of variability in application behavior. These are called trigger points..."); and an externalization component for providing the identified at least one point of variability with externalized business logic (see p. 22, §Trigger Point Framework – overview, "A trigger point is simply the location in a method of an object at which externalized business rules are invoked.", *Examiner interprets the Trigger Point Framework to be a component.*), the externalized business logic including an inferencing component (*see* above).

Regarding claim 33. *IBM* teaches a program storage device readable by a machine, tangibly embodying a program of instructions executable on the machine to perform method steps for managing a plurality of inferencing components (*see* p. 20, para. Firing Location, "... the rule implementor ... runs ... local to the trigger point (in the same JVM as the trigger point call). This would be on the client machine if the trigger point call is done there, or the server if the server part of an application makes a trigger point call."), the method steps comprising: identifying inferencing aspects for a program (*see* above); and providing the identified inferencing aspects as inferencing components (*see* above), wherein the inferencing components are externalizable (*see* above).

Regarding claim 2. *IBM* teaches the method of identifying inferencing aspects for a program wherein the providing step includes associating an externalized algorithm and data with each of the inferencing components (*see* p. 3, §BRBeans development and maintenance roles, "The application developer will typically assign the following to a rule: ... a default set of init parameters (the business rule values) (optional)", *Examiner interprets the "application developer" to develop an application that comprises and algorithm which calls rules which have parameters. Examiner interprets the value of rule parameters to be data.*).

Regarding claim 3. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the data is stored in persistent memory (see p. 6, §Database considerations, "There are several attributes in the BRBeans Rule EJB that may contain large amounts of data. ... The value

for these attributes is stored in a character type column within a database table.", *Examiner interprets a database to be persistent memory.*).

Regarding claim 4. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the identified inferencing aspects include at least one of a trigger point, a short term fact, an inference rule (*see* p. 2, §What is a business rule?, "A business rule is a statement that defines or constrains some aspect of a business ... At its simplest level, a business rule is little more than a well placed IF/THEN statement that compares a variable against a determined value, and then issues a command when they match.", *Examiner interprets the operation of an IF/THEN statement to be a modus ponens inference.*), an inference engine, a static variable mapping, a sensor, an effector, a long term fact, and a conclusion.

Regarding claim 5. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the inferencing components include at least one of a trigger point component, a short term fact component, an inference rule set component, an inference engine component (*see* pp. 29-31, §Rule Implementors interface – overview, *Examiner interprets RuleImplementor to be an inference engine.*), a static mapping component, a sensor component, an effector component, a long term fact component, and a conclusion component.

Regarding claim 6. *IBM* teaches the method of identifying inferencing aspects for a program; wherein each of the inferencing components is one of a consumer of data provided by an inferencing component, a supplier of data provided by an inferencing component, and a

combination thereof (*see* p. 32, §Externalized business rules, "When the fire() method is called on the Rule object, the Rule object combines its persistent set of values with the parameters it received on invocation to create the parameter list for the RuleImplementor ...", *Examiner interprets "the parameters ... received on invocation" to be data provided by an inferencing component (the trigger point framework) and the "persistent set of values" to be data supplied by the Rule object.*).

Regarding claim 7. *IBM* teaches the method of identifying inferencing aspects for a program, further comprising the step of associating at least one trigger point inferencing component with at least one application (*see* p. 23, § Determining where to place a trigger point, "To identify potential trigger locations in use case analyses such as this one, look for certain keywords such as: "if X is in a special category Y"", Examiner interprets the keywords to the antecedent to an inference of some potential component derived from use cases or user interaction scenarios. Examiner interprets the identification of the keywords to be associating at least one trigger point inferencing component with at least one application.).

Regarding claim 8. *IBM* teaches the method of identifying inferencing aspects for a program wherein trigger points operate either synchronously or asynchronously (*see* p. 30, "In some scenarios, the method fire may be called from multiple processes on the same instance of a RuleImplementor. This could happen if more than one client triggers the same rule at the same time. The implication of this is that the method fire should not change instance attributes of the implementor unless synchronization of the data being changed is performed.", *Examiner*

interprets "synchronization of the data being changed" for the method fire to be synchronous operation of a trigger point fire.)

Regarding claim 9. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the inferencing components is a master inferencing component that employs at least one other inferencing component (*see* p. 32, §Externalized business rules, "The RuleImplementor is a transient object (not managed by the application server) which the Rule instantiates and then uses to do the actual work.", *Examiner interprets the Rule to be the master inferencing component and the RuleImplementor to be at least one other inferencing component.*).

Regarding claim 10. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the inferencing components employs an inferencing engine (*see* pp. 29-31, §Rule Implementors interface – overview, *Examiner interprets RuleImplementor to be an inference engine.*).

Regarding claim 11. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the inferencing components is organized into at least one inferencing subcomponent (*see* pp. 45-46, §Writing your own strategies, "TriggerPoint object has its own set of strategies that can be changed independent of any other TriggerPoint object.", *Examiner interprets the "TriggerPoint object" to be an inferencing component and a strategy object to be an inferencing subcomponent.*).

Regarding claim 12. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the organization is one of an array (*see* p. 22, §Trigger Point Framework, "Build the array of objects containing the runtime parameters needed to satisfy the trigger point's business purpose."), a collection, a hashtable, an iterator, a list, a partition, a set, a stack, a tree, a vector, and a combination thereof.

Regarding claim 13. The method of identifying inferencing aspects for a program, wherein at least one of the inferencing components is composed of at least one inferencing subcomponent (*see* p. 21, § Dependent Rules, "Dependent rules can be nested within other dependent rules. In other words, a dependent rule of some particular rule can have its own dependent rules which, in turn, can have their own dependent rules, and so on.", *Examiner interprets nesting to be a form of composition and "dependent rules" to be "at least one" subcomponent of an inferencing compoment (rule).*).

Regarding claim 14. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the composition is one of an array, a collection, a hashtable, an iterator, a list, a partition, a set (*see* above, *Examiner assets that the set is the set of nested subcomponents.*), a stack, a tree, a vector, and a combination thereof.

Regarding claim 15. *IBM* teaches the method of identifying inferencing aspects for a program, wherein each of the inferencing components has at least one of an unique identifier, an intention,

a name (*see* p. 47, "Applications that use the BRBeans EJBs (this includes those that trigger rules or use the rule management APIs) must specify the JNDI names for these EJBs..."), a location, a folder, a start time, an end time, a priority, a classification, a reference, a description, a firing location, a firing parameter, and initialization parameter, an implementor, a ready flag, and free form data.

Regarding claim 16. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the inferencing components is shared by reference with at least one other inferencing component (see p. 4, §Why externalize rules?, "Reuse of rules across business processes. ...", *Examiner interprets "reuse of rules" to mean that at least one of the inferencing components is shared by reference with at least one other inferencing component.*).

Regarding claim 17. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the algorithms perform at least one of inferencing component creation (*see* p. 32, §Runtime behavior, "When the trigger point framework invokes fire on a Rule, it instantiates the RuleImplementor and uses it to do the actual work (to execute the rule algorithm or test). Once it has arrived at a result, the RuleImplementor returns that result. For constraint rules (ones that arrive at a boolean true/false answer) the returned value is, by convention, a ConstraintReturn. A ConstraintReturn is a data structure indicating whether or not the constraint was satisfied...", *Examiner interprets a ConstraintReturn to be an inferencing component bearing a truth assignment.*), inferencing component retrieval, inferencing component update, and inferencing component deletion.

Regarding claim 18. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the algorithms is shared by a plurality of inferencing components (see p. 27, §Using strategy objects to control triggers, "Default strategy objects are already defined for each of the four TriggerPoint steps ...", *Examiner interprets default strategy objects to comprise algorithms shared by the trigger methods: trigger(), triggerClassifier(), and triggerSituational().*).

Regarding claim 19. *IBM* teaches the method of identifying inferencing aspects for a program, wherein each of the algorithms is one of an execute trigger point algorithm, a return data algorithm (*see* p. 32, §Runtime behavior, "When the trigger point framework invokes fire on a Rule, it instantiates the RuleImplementor and uses it to do the actual work (to execute the rule algorithm or test). Once it has arrived at a result, the RuleImplementor returns that result. For constraint rules (ones that arrive at a boolean true/false answer) the returned value is, by convention, a ConstraintReturn. A ConstraintReturn is a data structure...", *Examiner interprets a ConstraintReturn to be the result of the execution of a return data algorithm.*), a join data algorithm, a filter data algorithm, a translate data algorithm, a choose by classification algorithm, a choose randomly algorithm, a choose round robin algorithm, an inference engine pre-processor, and inference engine post-processor, an inference engine launcher, a receive data algorithm, a send data algorithm, a store data algorithm, and a fetch data algorithm.

Regarding claim 20. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the providing step uses an inference component management facility to administer inferencing components, the administration including operations to create, retrieve, update, and delete (*see* p. 36, §Using the Rule Management Application (RMA) – overview, "The Rule Management Application (RMA) is a simple tool that assists the user in the high level administration of rules and rule folders. This includes the capability to create, modify, delete, import or export rules".).

Regarding claim 21. *IBM* teaches the method of identifying inferencing aspects for a program, wherein at least one of the inferencing components is composed of a plurality of inferencing subcomponents (*see* p. 21, §Dependent Rules, "Dependent rules can be nested within other dependent rules. In other words, a dependent rule of some particular rule can have its own dependent rules which, in turn, can have their own dependent rules, and so on.", *Examiner interprets "nesting" to be a form of composition.*).

Regarding claim 22. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the composition occurs one of statically (*see* p. 21, §Dependent Rules, "Dependent rules are specified in the attributes of the top-level rule where the fully-qualified name of each dependent rule is listed.", *Examiner interprets specification "in the attributes of the top-level rule" to be a static compossition.*), dynamically, and a combination thereof.

Regarding claim 23. *IBM* teaches the method of identifying inferencing aspects for a program, wherein the composition occurs using an inference component management facility (*see* pp. 36-37, §Creating rules, "In the New Rule properties window, use the following tabs to define the rule: ... Dependent Rules: Use this tab to specify the rules that the newly created rule will depend upon.", *Examiner interprets the "Rule Management Application" to be a inference component management facility.*).

Regarding claim 25. *IBM* teaches the system for providing externalized business logic, wherein the inferencing component includes an externalized algorithm and data (*see* p. 32, §Runtime behavior, "...the trigger point will assemble the data that will be sent as parameters to each Rule. ... When the trigger point framework invokes fire on a Rule, it instantiates the RuleImplementor and uses it to do the actual work (to execute the rule algorithm...", Examiner interprets a Rule object to be an inferencing component and the parameters as data.).

Regarding claim 26. *IBM* teaches the system, further including a persistent memory component configured to persistently store the data (*see* p. 6, §Database considerations, Examiner interprets relational database management systems supported by BRBeans Rule EJBs to be a persistent memory component configured to persistently store the data.).

Regarding claim 27. *IBM* teaches the system for providing externalized business logic, further including an execution component for executing the externalized algorithm using at least one virtual machine (*see* p. 44, "Another way that performance can be improved is to have all

"client" code that triggers rules run in a servlet. This assumes that the servlet is running on the same physical system as the EJB server where the BRBeans EJB are installed. This way when remote calls are made to the EJB rule server, they are going to another JVM on the same machine", *Examiner interprets JVM (Java Virtual Machine) to be the at least one virtual machine which executes the externalized algorithm of a triggered rule on the same physical system as the EJB server.*).

Regarding claim 28. *IBM* teaches the system, wherein the inferencing component is composed of a plurality of inferencing subcomponents (*see* p. 21, §Dependent Rules, "Dependent rules can be nested within other dependent rules. In other words, a dependent rule of some particular rule can have its own dependent rules which, in turn, can have their own dependent rules, and so on.", *Examiner interprets "dependent rules" to be inferencing components.*).

Regarding claim 29. *IBM* teaches the system, wherein the composition occurs dynamically (*see* p. 22, §Trigger Point Framework – overview, "2. Build the array of objects containing the runtime parameters needed to satisfy the trigger point's business purpose. This array is normally passed as one of the parameters of the fire method of the RuleImplementor.", *Examiner interprets the array of objects to comprise "dependent rules".*).

Regarding claim 30. *IBM* teaches the system, wherein the composition occurs statically (*see* p. 21, §Dependent Rules, "Dependent rules are specified in the attributes of the top-level rule where

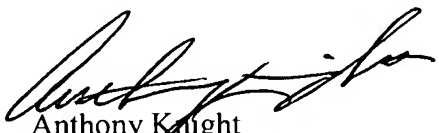
the fully-qualified name of each dependent rule is listed.", *Examiner interprets specification "in the attributes of the top-level rule" to be a static compossition.*).

Regarding claim 31. *IBM* teaches the system, wherein the composition occurs in part dynamically and the remainder statically (see p. 21, §Dependent Rules, "Dependent rules are specified in the attributes of the top-level rule where the fully-qualified name of each dependent rule is listed. When the top-level rule is triggered, an array of dependent rule names is passed to the rule implementor's initmethod. They are stored here until they are triggered by the fire method.", *Examiner interprets this procedure to work in the case of a dependent rule, statically defined for a non-dependent rule, which is composed of other dependent rules, such that the statically defined dependent rules' dependent rules are dynamically init'ed (i.e. composed for the dependent rule) by the rule implementor's initmethod after the non-dependent rule is fired and after its top level dependent rule is fired.*).

Regarding claim 32. *IBM* teaches the system, wherein the identified at least one point of variability includes at least one of a trigger point, a short term fact, an inference rule (see *see* p. 2, §What is a business rule?, "A business rule is a statement that defines or constrains some aspect of a business ... At its simplest level, a business rule is little more than a well placed IF/THEN statement that compares a variable against a determined value, and then issues a command when they match.", *Examiner interprets the operation of an IF/THEN statement to be a modus ponens inference.)*, an inference engine, a static variable mapping, a sensor, an effector, a long term fact, and a conclusion.

Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nathan H. Brown, Jr. whose telephone number is 571-272- 8632. The examiner can normally be reached on M-F 0830-1700. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Anthony Knight can be reached on 571-272-3687. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



Anthony Knight
Supervisory Patent Examiner
Tech Center 2100

Nathan H. Brown, Jr.
September 26, 2006